IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

SmartBoost: A Power Migration Mechanism Leveraging On-chip Caches

Journal:	*Transactions on Computer-Aided Design of Integrated Circuits and Systems*
Manuscript ID	TCAD-2017-0276
Manuscript Type:	Full Paper-System-Level Design
Date Submitted by the Author:	11-Jul-2017
Complete List of Authors:	Zhou, Zhe; Peking University, Computer Science Zhang, Chao; Peking University, EECS; Sun, Guangyu; Peking University, Computer Science
Keywords:	Power Migration, DVFS, ILP, CMP

Cover Letter

Dear Editor,

We would like to submit the enclosed manuscript entitled “SmartBoost: A Power Migration Mechanism Leveraging On-chip Caches”, which we wish to be considered for publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

We believe that two aspects of this manuscript will make it interesting to general readers of your journal. First, we analysed both the cores and the caches to improve runtime-performance, while existing DVFS method only focused on cores. A comprehensive analysis means that we could maximize performance of processors. Second, we explored a run-time ILP(Integer Linear Programming) method to ensure the power was allocated to where it was most needed. Thus, our mechanism achieved a pioneering comprehensive and optimum power migration.

The work described has not been submitted elsewhere for publication, in whole or in part, and all the authors listed have approved the manuscript that is enclosed.

Thank you very much for your time and consideration.

Sincerely yours,

周喆
Peking University, Computer Science.
Phone: +86 15201116415
Email address: zorichzorich@gmail or 15201116415@126.com

SmartBoost: A Power Migration Mechanism Leveraging On-chip Caches

Zhe Zhou[1], Chao Zhang[2], Guangyu Sun[3], Zhen Tan[4], Qinyuan Wei[4], and Jia Di[4], *Fellow, IEEE*
[1]Center for Energy-Efficient Computing and Applications, Peking University, Beijing, China

Dark silicon challenge is becoming ever severe with unstoppable increasing number of integrated transistors. Traditional power management schemes focusing on reducing power consumption have to be pivoted to the counterpart target: Optimize the performance within limited power. However, the power of current processors is frequently wasted among their on-chip caches, which consume large chip area but deliver insufficient performance benefit. The efficiency of processor power should by improved in the dark silicon era.

To solve this problem, we propose SmartBoost, a power migration mechanism based on standard DVFS, dynamically reallocating power supply to the most beneficial components of a processor, to improve the energy efficiency. SmartBoost is a pure hardware solution which pioneers activity prediction and integer linear programming (ILP) which can achieve the optimal solution theoretically. Applied on a multi-core processor without existing DVFS, the SmartBoost successfully increases the performance by 22.9% **on average. Compared with standard DVFS, the power consumption of SmartBoost with look-up table is reduced by** 15.1%**.**

***Index Terms*—Power Migration, DVFS, ILP, CMP.**

I. Introduction

POWER consumption is a critical design issue for modern processors. To balance the performance and the power cost, a huge number of power management schemes have been proposed and applied in physical processor implementations. These approaches handle the power problem from two aspects: (a) Dynamical power supply, such as dynamic voltage or frequency scaling (DVFS) [1], [2], [3], [4], power or clock gating [5], [6], and switching on/off caches [7], [8], [9], [10] etc. (b) Statical low power status, such as near-threshold computing [11], [12], [13] and non-volatile memory techniques [14], [15], [16], [17]. The goal of these approaches is to try to reduce power consumption with the assumption that there exists *power surplus*. However, such an assumption may not always be valid for modern processors.

In fact, modern processors usually encounter problems of power shortage rather than power surplus in the dark silicon era. On the one hand, since the number of transistors integrated on a single chip keeps increasing, the demand for on-chip power consumption keeps increasing. The circuit supply voltage and transistor threshold voltage, however, cannot scale proportionally to prevent growth in on-chip power density [18], [19]. On the other hand, the exacerbation of voltage fluctuation problem requires more power to be supplied to reach enough design margins. Unfortunately, the power-shortage problem gets even worse due to the fact that the density of chip package pins for power supply is expected to remain unchanged in the near future [20]. In other words, we have entered the era of dark silicon that there lacks enough power to boot up all components inside CPU [18], [19]. In fact, less than 20% of transistors can be activated according to the projection [21].

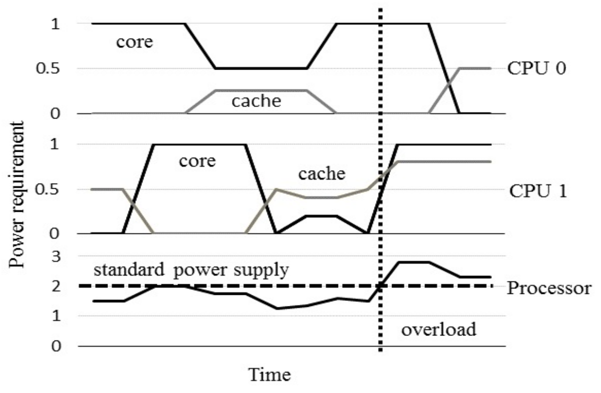

Fig. 1. An illustration of the unbalance within a dual-core processor.

Consequently, state-of-the-art power management is trying to fully utilize the limited power supply. Xeon E5-2600 [22] supports inter-core dynamic voltage and frequency scaling within multiple supply voltages. Over-clocking techniques [23], [24] are widely used to elevate the power status of the cores. In order to further improve the energy efficiency, Snatch [25] leverages the unbalance between the processor and main memory. However, the method to improve the power efficiency within the processor among different components is still not well explored.

Shown in Fig. 1, the power consumption of cores and caches is not balanced. Even though this leaves the room for better energy efficiency, there are two major challenges to solve: (a) Predict the activity of various components without the interference of operating system; (b) Generate the runtime power allocation decisions considering the behavior of both cores and caches.

In this paper, we propose SmartBoost, a on-chip power migration method, to improve the energy efficiency of a processor. SmartBoost is an improved version of standard DVFS and inherits its basic dynamic voltage and frequency scaling idea. However, different from standard DVFS, SmartBoost takes the whole processer into account(not only the cores). Moreover, SmartBoost has a predictor to predict the behavior of different components with in the chip. According to the prediction, SmartBoost uses runtime integer linear programming (ILP) to migrate the effective power supply within the processor from the components delivering less performance to the ones delivering more. SmartBoost could also avoid runtime ILP calculating by look-up table, which could reduces hardware and power cost further.

The contributions of this paper are summarized as follows:

- An improved power migration method based on DVFS is introduced, targeted on maximizing the performance with power constraints.
- The relationships between power and voltage are simplified and verified to support the runtime power management.
- Predictors based on polynomial functions are proposed for the prediction of processor activity.
- Two power allocation methods are introduced: runtime integer linear programming (ILP) and look-up table method.
- The latency of ILP solver and look-up table (LUT) is explored and proved fast enough for on-chip power management.

II. Design of SmartBoost

The target of SmartBoost is to optimize the performance limited by power according to the projection of activity, depending on solving integer linear programming problem at runtime. Different from standard DVFS, caches are also considered as individual components in the power management. This section introduces the design of activity predictor, ILP model, optimization, power migration policy and the overhead of our designs.

A. Projection of Activity

The activity is quantified to facilitate the power management. It indicates the activity level of a component on chip in a time period. The activity of core is defined as the actual instruction amount per sample period normalized with its maximum value; while the activity of cache uses the access amount per sample period. The activity of the processor can be represented as an activity vector $\boldsymbol{A} = (a_0, a_1, ..., a_n)$, where $a_i \in [0,1]$.

The activity history is used to predict the activity vector in next sample period. We propose two prediction methods: polynomial prediction and moving average prediction. Polynomial prediction fits a linear/quadratic/cubic curve, using two/three/four history records. Using the curve, the next record can be predicted. The second method calculates the exponential moving average(EMA) of the history activity. The recent record is assigned a doubled weight than the its previous one, which is shown in (1).

$$\boldsymbol{A}_n = \frac{2^{k-1} \times \boldsymbol{A}_{n-1} + 2^{k-2} \times \boldsymbol{A}_{n-2} + ...1 \times \boldsymbol{A}_{n-k}}{2^{k-1} + 2^{k-2} + ...1} \quad (1)$$

The prediction accuracy of those methods is tested by PARSEC benchmark in Table I. The relative differences between the "real" and the predicted activity value for all predicted records are averaged to calculate the prediction accuracy. *HistoryRecord* in the table indicates how many history activity records are used by that prediction method. The quadratic polynomial prediction needs at least 3 points to determine a quadratic curve; and the cubic polynomial needs 4 records. The '-' sign means more records are necessary for that prediction method.

The linear polynomial using 2 history activity records shows the best prediction accuracy. Higher level of polynomial prediction does not show higher accuracy, because the sudden variation of the activity shows very little predictability if no operating system support is provided. Thus the linear polynomial prediction method is chosen in SmartBoost. The average prediction accuracy is 79.26%, shown in the Fig. 2.

TABLE I
Prediction Accuracy of Different Methods

	History Record		
Prediction	2	3	4
EMA	77.70%	77.75%	77.50%
Linear Poly	**79.26%**	78.66%	77.82%
Quadratic Poly	-	67.35%	69.65%
Cubic Poly	-	-	61.76%

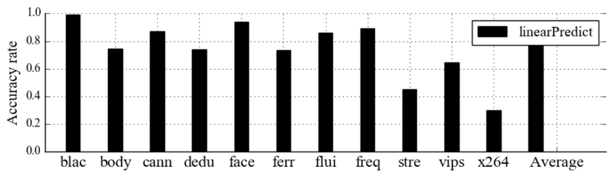

Fig. 2. Prediction accuracy of different benchmarks using linear fit prediction

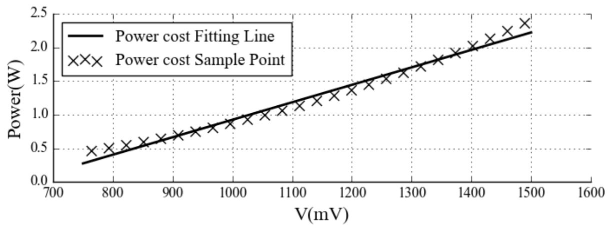

Fig. 3. Scatter plot and fitting curve between power and voltage.

B. Integer Linear Programming

In this section, we focus on how the ILP works. Voltage contribution of n major components(cores and caches) is outcome of ILP, which is represented by a n-length vector $\boldsymbol{V}$. It is the variable for the ILP. The main constraint of ILP is power limitation, and the objective function is performance gain. Another n-length vector $\boldsymbol{A}$ is used to represent the activity of components within the processor, which is mentioned before. To implement an efficient run-time power migration policy, the value of $\boldsymbol{A}$ and $\boldsymbol{V}$ is discretized to three levels, respectively. After the preparing work, we now explain the linear equation needed by ILP.

1) Power Constraint Equation

The power cost constraint equation is represented as (2). P_{sepc} is the power limitation, P_{cost} is the power cost we evaluated.

$$P_{spec} \geq P_{cost} \quad (2)$$

Linear fitting between voltage ($\boldsymbol{V}$) and power (P_{cost}) for ILP constraint is shown in Fig. 3. The data are based on a previous technical report [26]. Since the data reported are based on $0.18\mu m$ technology node, we scale it to $45nm$ node ARM processor to make them compatible with other data collected from CACTI [27]. Within in the $[750, 1500]mV$ voltage range, the relation can be approximated by a linear function without losing too much accuracy.

$$P_{spec} \geq P_{one} = V \cdot K_{one} + B_{one} \quad (3)$$

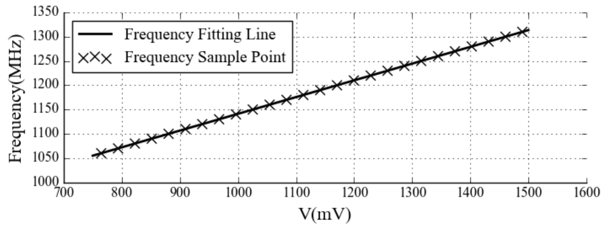

Fig. 4. Scatter plot and fitting curve between frequency and voltage.

Power constraint equation of one component is represented as (3). The P_{one} indicates power of one component. The K_{one} and B_{one} are the first-order and constant coefficient of linear fitting between voltage and power, respectively.

With different activity level, the power of component would change in equal proportion. For runtime, power cost constraint equation of all components with different activity is shown as (4). The $\boldsymbol{I}$ is the n-dimensional unit vector. To make the linear relationship between voltage and power more clear, the $\boldsymbol{K}_p$ is defined to indicate $K_{one} \cdot \boldsymbol{A}$.

$$\begin{aligned} P_{spec} &\geq \boldsymbol{V}^T(K_{one} \cdot \boldsymbol{A}) + \boldsymbol{A}^T(B_{one} \cdot \boldsymbol{I}) \\ &= \boldsymbol{V}^T \cdot \boldsymbol{K_p} + \boldsymbol{A}^T(B_{one} \cdot \boldsymbol{I}) \end{aligned} \tag{4}$$

2) Performance Objective Function

The objective function is used to describe the target of the ILP. SmartBoost uses gain (reduction of execution time) to indicate the performance. The performance gain consists of the gain of cores (Q) and caches (H), which is shown in (5). The influence of interact between caches and caches is omitted here. Which component of chip need more power is a priority issues and does not need exorbitant precision.

$$G = Q + H \tag{5}$$

The gain of a core (Q_{one}) is based on the time consumed in the core pipeline, which is linear related to the core frequency. The gain is calculated by (6), where the t_{inst} is the average time consumed by each instruction, and f_o, f are the core frequencies before and after the voltage change, respectively.

$$Q_{one} = N_{inst} \cdot t_{inst} \cdot f_o \cdot (\frac{1}{f_o} - \frac{1}{f}) \tag{6}$$

Actually, the relationship between the maximum switching frequency and supply voltage is fitted by a linear function in Fig. 4.Within the fitting period of voltage from $0.75V$ to $1.5V$, the R square is 0.99. According to this fitting model, an ARM core working at 1V can run at about 1.1GHz. The relationship between frequency and voltage is shown in (7).

$$f = V \cdot K_f + B_f; \tag{7}$$

$$Q_{one} \approx \alpha \cdot Activity \cdot t_{inst} \cdot \frac{K_f(V - V_o)}{f_o} \tag{8}$$

To get a linear relationship, the Q_{one} is expanded in a Taylor series and we choose the linear terms of V in the second line of the (8). The N_{inst} is expressed as $\alpha \cdot Activity$, the value of α should be the instruction number of the core whose activity is 1. The K_f has mentioned in (7). The V_o is set as $1.1V$ in our evaluation platform. In this case, the deviation of our fitting is less than 0.3% when the voltage is in range $0.75V$ to $1.5V$.

Performance gain of all cores with different activity is shown in (9). The $\boldsymbol{A}_{core}$ is cores part of Activity vector ($\boldsymbol{A}$). The $\boldsymbol{I}$ also represent the unit vector. To make the linear relationship more clear, the $\boldsymbol{K}_{core}$ is defined to indicate $(\alpha \cdot t_{inst} \cdot K_f)/(f_o) \cdot \boldsymbol{A}_{core}$.

$$\begin{aligned} Q &= \frac{\alpha \cdot t_{inst} \cdot K_f}{f_o} \cdot \boldsymbol{A}_{core} \cdot (\boldsymbol{V}_{core} - V_o \cdot \boldsymbol{I}) \\ &= \boldsymbol{K}_{core} \cdot \boldsymbol{V}_{core} + \boldsymbol{K}_{core} \cdot (-V_o \cdot \boldsymbol{I}) \end{aligned} \tag{9}$$

The gain achieved by caches (H_{one}) comes from the reduction of cache misses. To increase the accuracy of approximation, only L2 and L3 caches will be turned off or slowed down, which will not stall the memory stage of pipeline due to fetch miss during power off. Previous work [9], [10], [17] shows that the L2/L3 access latency (including hit and miss) can only be reflected to the overall execution time by a factor. Thus the performance gain of turning on a cache can be estimated by (10). The β is the constant used to represent the core's cache latency cover effect and the overhead due to loss of data. Dirty data within the cache may be lost, if the cache is shutdown entirely. Thus, eager data write back policies and non-inclusive cache hierarchy is used to keep the data integrity. In this way, the potential data coherency issue is solved. But the overhead of loading data back also reduces the gain. Considering all the overhead, we set the β to 0.1 in our model.

$$\begin{aligned} H_{one} &= \beta \cdot N_{access} \cdot (t_{miss} - t_{hit}) \\ &\approx \beta \cdot \alpha \cdot Activity \cdot period \cdot \frac{K_f(V - V_o)}{f_o^2} \end{aligned} \tag{10}$$

The $t_{miss} - t_{hit}$ is also inversely proportional to f which has a linear relationship with V. Thus, the H_{one} is also expanded in a Taylor series to find approximate linear relationship between H_{one} and V. The constant *period* indicates the time span of a period. The N_{access} is expressed as $\alpha \cdot Activity$. V_o is set as $1.1V$. In this case, the deviation of our fitting is less than 1% when the voltage(V) is in range $0.75V$ to $1.5V$.

Performance gain of all caches with different activity is shown in (11). To make the linear relationship more clear, the $\boldsymbol{K}_{cache}$ is defined to indicate $(\beta \cdot \alpha \cdot period \cdot K_f)/(f_o^2) \cdot \boldsymbol{A}_{cache}$.

$$\begin{aligned} H &= \frac{\beta \cdot \alpha \cdot period \cdot K_f}{f_o^2} \cdot \boldsymbol{A}_{cache} \cdot (\boldsymbol{V}_{cache} - V_o \cdot \boldsymbol{I}) \\ &= \boldsymbol{K}_{cache} \cdot \boldsymbol{V}_{cache} + \boldsymbol{K}_{cache} \cdot (-V_o \cdot \boldsymbol{I}) \end{aligned} \tag{11}$$

3) ILP Equation and Optimization

Based on above simplification, The power management problem is converted to an integer linear programming (ILP) problem. SmartBoost would deliver this problem to a hardware ILP solver. The restrictions and object function is shown in (12). The V_{min} and V_{max} is predetermined value to restrict the voltage range. In our experiment, the voltage range is $[0.75V, 1.5V]$.

$$\begin{aligned} maximize \quad & (\boldsymbol{K}_{core} \cdot \boldsymbol{V}_{core} + \boldsymbol{K}_{cache} \cdot \boldsymbol{V}_{cache}) + \\ & \boldsymbol{K}_{core} \cdot (-V_o \cdot \boldsymbol{I}) + \boldsymbol{K}_{cache} \cdot (-V_o \cdot \boldsymbol{I}) \\ subject \quad to \quad & \boldsymbol{V}^T \cdot \boldsymbol{K_p} + \boldsymbol{A}^T(B_{one} \cdot \boldsymbol{I}) \leq P_{spec} \\ and \quad & \boldsymbol{V}_i \geq V_{min}, \quad \boldsymbol{V}_i \leq V_{max} \end{aligned} \tag{12}$$

To reduce the hardware overhead of ILP solver, SmartBoost could uses look-up table instead of hardware ILP solver. the value in table are calculated by ILP beforehead. Of course, when there are too many cores and caches in the chip, this method does not work well for it needs huge memory space.

C. SmartBoost Power Migration

Using an ILP solver, the SmartBoost power migration is implemented by the Algorithm 1. The SmartBoost works in parallel with the other components of the chip. The algorithm can be divided into two parts. (a) Line 1-2: The coefficients are initialized in factory, according to the configuration and attributes of the processor. (b) Line 3-18: During the k^{th} sampling period, the activity of all components are predicted according to their history(SmartBoost is work parallel with the k^{th} period, thus SmartBoost only have history record of $(k-1)^{th}$ and $(k-2)^{th}$ period), and the $\boldsymbol{K}_p$, $\boldsymbol{K}_{core}$, $\boldsymbol{K}_{cache}$, are calculated individually. At last the $\boldsymbol{V}$ is produced by the ILP solver or look-up table.

Algorithm 1 SmartBoost Power Migration Algorithm

```
1:  Initialize $K_{one}, B_{one}, \alpha, t_{inst}, K_f, f_o, V_o, \beta, period$
2:  $\boldsymbol{A_0} \leftarrow 1, \boldsymbol{V} \leftarrow 1$.
3:  for each sampling period $k$ do
4:      Initialize $\boldsymbol{K}_p$, $\boldsymbol{K}_{core}$, $\boldsymbol{K}_{cache}$ to 0.
5:      $\boldsymbol{A}_{k+1} \leftarrow$ActivityPredictor($\boldsymbol{A}_{k-1}, \boldsymbol{A}_{k-2}$)
6:      $\boldsymbol{K}_p \leftarrow K_{one} \cdot \boldsymbol{A}_{k+1}$
7:      for each component $i$ do
8:          if component $i$ is a core then
9:              $\boldsymbol{K}_{core} \leftarrow (\alpha \cdot t_{inst} \cdot K_f)/(f_o) \cdot \boldsymbol{A}_{k+1}[i]$
10:         else if component $i$ is a cache then
11:             $\boldsymbol{K}_{cache} \leftarrow (\beta \cdot \alpha \cdot period \cdot K_f)/(f_o^2) \cdot \boldsymbol{A}_{k+1}[i]$
12:         end if
13:     end for
14:     $\boldsymbol{V} \leftarrow$ILPSolver(or look-up table)($\boldsymbol{K}_p$, $\boldsymbol{K}_{core}$, $\boldsymbol{K}_{cache}$)
15: end for
```

D. Design Overhead

To reduce overhead of activity predictor, SmartBoost uses the linear polynomial prediction method. According to the design compiler, the predictor of one component of the chip is not larger than 2000 transistors. For the 4-core ARM chip used in our evaluation, it is divided into 9 components: 4 cores (includes L1 I/D caches), 4 private L2 caches and one shared L3 cache. According to the design compiler, total hardware overhead of the predictor is less than 20×10^3 transistors. For each predictor of the component, a 2-word space is used to record the history activity level, thus we need $18 \times 64 - bit = 1152 - bit$ space.

In order to make the ILP solver fast enough for runtime decision making, a hardware ILP solver proposed by previous work [28] is employed. The scale of the ILP problem is estimated by the product of the number of variables and the number of constraint equations. The voltage vector is 9 units in length. And the upper and lower bounds of the voltage contribute to 18 constraints. In short, the ILP problem of a 4-core processor has a 9×19 scale. We use 2×2 block size and $12 - bit$ word-length ILP hardware design proposed in previous work [28]. According to the design compiler, the hardware overhead of ILP is about 2556×10^3 transistors.

TABLE II
THE OVERHEAD OF SMARBOOST

	Transistor	Memory (Kb)	Area (mm^2)	Power (mW)	Time (us)
Predictor	20×10^3	2	0.0025	1.1	0.001
ILP Solver	2556×10^3	72	0.3232	136.9	10
look-up table	1×10^3	314	0.2917	0.1	0.001
Overall(ILP solver)	2576×10^3	74	0.3257	138.0	10.001
Overall(look-up table)	2557×10^3	316	0.2942	1.2	0.002

There are a $18 - Kbit$ RAM in each block, thus the ILP hardware solver need $72 - Kbit$ RAM. The time cost of the ILP solver is less than $10us$.

To avoid the cumbersome ILP hardware solver, SmartBoost could replace hardware solver by a look-up table. The index of table is activities of components in chip and the content is the voltages of components. For each component, there are 3 activity levels and 3 voltage levels. For a 9-length activity vector, $\log_2(3^9) \leq 16 - bit$ memory space is need. Thus, SmartBoost with look-up table need $3^9 \times 16 - bit = 314 - Kbit$. According to CACTI 5.3, the look-up table would take $0.2917mm^2$ area. The time overhead would less than $1us$, it also need less than 1×10^3 transistors.

Evaluated at 22nm CMOS processing node, the detailed overhead analysis is shown in Table II. The span of period is $100us$. The overall time needed of SmartBoost with ILP solver is $10us$(SmartBoost with table is even faster), thus SmartBoost could give the chip a voltage distribution of next period in time.

III. EVALUATIONS

The performance and power consumption using SmartBoost are evaluated in this section.

A. Experimental Setup

We evaluate the SmartBoost based on the gem5[29] simulation framework. The processor has 4 core, 4 private L2 caches and a shared L3 cache. Each of them is assigned a voltage domain to facilitate the power management.The detailed parameters can be found in Table III.

Targeting on shared-memory programs for multiprocessors, PARSEC [30] benchmark suite are executed (except 'rtview' and 'vips') in our evaluations. Benchmarks are abbreviated labeled by their first 4 characters in this section. All benchmarks are detailed executed for 100 milliseconds after fast-forwarding the beginning segments, which is enough to evaluate the proposed power migration framework. The statistics and traces are collected periodically for profile and run-time analysis. The statistics from gem5 are analyzed by McPAT [31] to generate the data traces of power consumption. The voltage information is obtained from VoltSpot [18] by inputing the power traces. Both McPAT and VoltSpot are validated against the data of commercial processors, and are deliberately used in this work considering the model error cancel [32].

B. Run-time Traces

As mentioned, programs generally have executing phases that show the unbalance of power consumption among different components. The activity traces and the SmartBoost

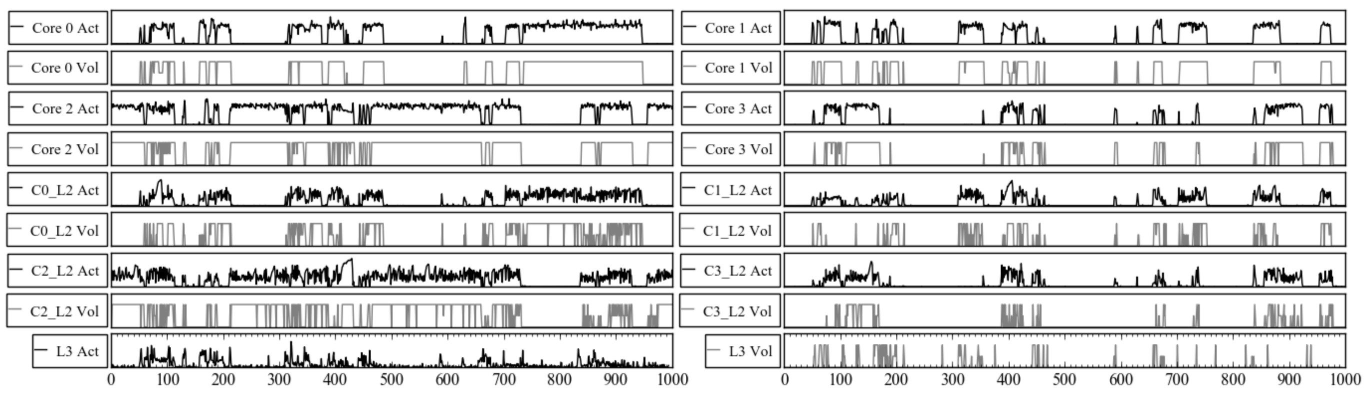

Fig. 5. The variation traces of activity and SmartBoost decisions

TABLE III
SYSTEM CONFIGURATION

	Configurations
Core	4 cores, ARM A15 Out-of-Order, 1.0 GHz
L1 I/D	32KB, private, 2-way, 64B, LRU, 4 MSHRs, 8 WBs
L2	2MB, Private, 4-way, 64B, LRU, 20 MSHRs, 8 WBs
L3	16 MB, Shared, 16-way, 64B, LRU, 20 MSHRs, 16 WBs
Memory	512MB, DDR3-1600, 12.8GB/s, avg. lat. 30ns
voltage	0.75V-1.5V

decision traces made on voltage of a representative benchmark ("dedup") is shown in Fig. 5. The symbol "CoreN" indicates the normalized activity for the N^{th} core, while the L2_N is the private L2 cache for the core. The activities of caches and cores are normalized to their maximum activities, obeying the activity definitions in the proposed framework. The traces include 1000 equal-length periods, whose duration is $100\mu s$. Shown by the activity traces, the program has execution phases: During periods 250 to 300, only one core is active, but the caches are sometimes busy.

Tracking the activity of all the components, SmartBoost power migration framework decides whether the cores will be speed up or the caches will be turned off or slowed down. All voltage values are normalized to their maximum values. The value '0' for a core indicates the core is working at its lowest frequency, while the '0' of a cache indicates that the cache has been turned off using power gating techniques. During the 250-300 region, only the core 2 is very active, thus the other components are in low voltage modes, when the power is migrated to the core 2 and the L3 cache. During the periods from 400 to 450, four cores are all active. The SmartBoost maintains the voltage of L3 cache at low levels, to migrate power to cores. The voltage statuses of private L2 caches are consistent with their cores. Thus, in order to maximize the performance of the entire processor, SmartBoost sets the caches usually at low-voltage level, migrating the extra power to cores.

C. Performance and Power Consumption

The results shown in Fig. 6 estimate the speedup within a 4-core ARM processor. The performance is calculated by taking average of the time used by the four cores. All performance speedups are compared with a baseline processor without

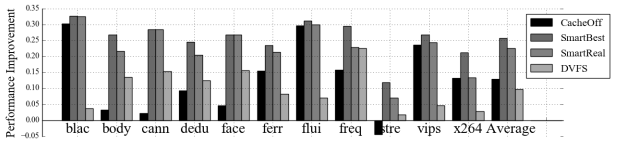

Fig. 6. Performance improvement for a 4-core ARM processor.

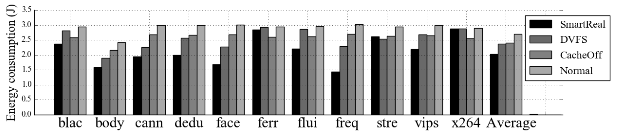

Fig. 7. Energy consumption for a 4-core ARM processor.

SmartBoost or DVFS. CacheOff indicates switching off L2 and L3 cache and providing the rest power to cores. CacheOff has a nice performance improvement in some benchmark ("blackscholes" and "fluidanimate"), and have a 18.4% performance improvement in average. However, it does not work well in 'streamcluster' because caches indeed provide lot of benefit in this case.

As we mentioned, SmartBoost includes prediction and optimization. If SmartBoost has 100% accuracy rate in prediction, the optimization would give an outcome which is nearing the optimal solution. The "SmartBest" labels this optimal solution, while "SmartReal" labels the actual results. In some benchmark ("blackscholes", "facesim"), the "SmartReal" permits a very close approach to the "SmartBest", thanks to the high prediction accuracy. However, in benchmark "freqmine" or "x264", the gap between "SmartBest" and "SmartReal" still exists. In those case, the behaviors of the cores and caches are hard to predict through pure hardware method. According to the Fig. 2 in Section II-A, the prediction miss in benchmark "blackscholes" is negligible and the predictor performs not well in benchmark "x264", too. On average, the performance of 'SmartBest' can be improved by 26.6% and the performance of "SmartReal" can be improved by 22.9%.

To investigate the energy efficiency, we run the same amount of work in SmartBoost, DVFS, cache-off and the baselines.The energy consumption of cores and caches(not include SmartBoost itself) in different benchmarks is shown in Fig. 7. DVFS does do well compared to the baselines,

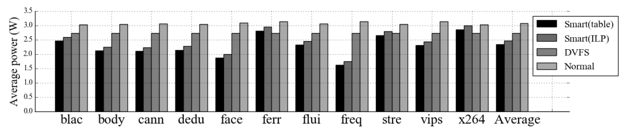

Fig. 8. Average power for a 4-core ARM processor.

TABLE IV
COMPARISON

	advantage	disadvantage
SmartBoost (look-up table)	high performance; less power than ILP	limited number of components in chip;
SmartBoost (ILP solver)	high performance	high design difficulty; high design and power cost
standard DVFS	reliable	do worse in power shortage; could not optimise caches
Cache-Off	easy to design	do worse in data-intensive program

which shows a significant effective energy consummation for benchmark "ferret", whose processor is blocked by hard drive accesses for a long time. In this case, the DVFS is nearly as efficient as SmartBoost, since the entire processor is blocked and idle. For the benchmarks (e.g. freqmine) that only the caches are also active, DVFS does worse than SmartBoost, since the voltage scaling of DVFS is processor-level. On the other hand, although cache-off always reduces the energy cost, SmartBoost does even better. The cache-off reduces the power of chip by reducing the voltage of cache, however it will increase the run time, thus consume more energy. In average, the energy consumption of SmartBoost is 25.4% less than normal case.

We also compare the average power of whole chip(include SmartBoost itself) shown in Fig. 8. In average, the power efficiency of SmartBoost with ILP is 19.5% higher than normal case. It has a 10.4% efficiency improvement than DVFS. The SmartBoost with look-up table uses 15.1% power less than DVFS.

D. Comparision

The Comparison between SmartBoost and other power migration methods is shown in Table IV.

IV. Related Work

Among amounts of literature and commercial processors, improving processor power performance by temporarily exceeding the thermal design power (TDP) is a very attractive topic. Increasing the speed (boost) of a processor in a short time period is such an attractive feature that both Intel and AMD have introduced turbo techniques [33], [24]. These boosting techniques save power for several cores using processor-wide power states. However, these techniques are limited by the coarse processor-level power management, which cannot leverage the power phase changes within a core efficiently. To tolerant more heat in a short time period, computational sprinting [34] is proposed to speedup applications by exceeding the thermal design power (TDP) in a short time period. It thoroughly considers the PDN designs and the thermal packaging issues. However, this method assumes the program can be ideally split into multiple parallel threads and execute on a multi-core processor. Limited by the methodology, the performance of single thread program cannot be improved. On the other hand, exceeding the TDP requires extra bulk materials or phase change materials to store the heat, which induces more complexity in heat sink designs. To leverage the power changing more efficiently, Gupta et al. [35] proposed Tribeca to dynamically adjust voltages and frequency margins. It studies the parameter variations including process variation, voltage variation, and temperature noises. They investigated fin-grained global frequency tuning and per unit voltage tuning. However, none of the techniques above can improve the single-core processor performance, via better power allocation within a processor.

Using on-chip power efficiently also raises the opportunity to trade reliability for performance. Previous work trades some power supply pins for a wider data path. Zhang et al. [18] proposed a pre-RTL model for PDN, to study the trade-off between power supply and performance. Chen et al. [36] proposed switchable pins to increase the off-chip data path bandwidth, to boost the core performance. It dynamically switches several chip pins used for power supply to data I/O. These studies suggest the ability to tolerant power shortage can further benefit the performance.

As far as we know, we are the first work to show on-chip power migration from on-chip caches to cores and to improve the performance and reliability. Future work can further enlarge the benefit of power migration by the follow two points: (a) Different clock rates between core and its private cache could reduce data transfer efficiencies, while a new buffer mechanism maybe reduce this loss. (b) There is a gap between SmartReal and SmartBest which main problem is prediction accuracy. Thus, a operation system support maybe help SmartBoost.

V. Conclusion

As we have entered the era of Dark Silicon, the pivot of power management is how to utilize the limited on-chip power efficiently. Unfortunately, current power management schemes, such as standard DVFS, may not achieve this goal for two reasons. First, they are proposed to reduce power under performance constraints rather than improving performance under power constraints, especially when a power shortage happens. Second, they mainly focus on coarse-grain core-level power management, and each core is managed individually. Thus, the issue of unbalanced inter-core and intra-core power demand is not well addressed.

To overcome these problems, the SmartBoost is proposed in this work. It can reallocate limited power from the perspective of the whole processor to achieve the best energy efficiency. Especially when a power shortage happens, it can dynamically migrate power to the components with highest performance gain.With the help of SmartBoost, we are able to increase the performance of a multi-core processor up to 32.6% (22.9% on average), and increase the energy efficiency by 19.5%. Compared with the conventional DVFS achieving the same performance of SmartBoost, SmartBoost with look-up table reduces the power consumption by 15.1%.

References

[1] W. Kim, M. S. Gupta, G. Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," *Proceedings*

- *International Symposium on High-Performance Computer Architecture*, pp. 123–134, 2008.
[2] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," in *37th IEEE International Conference on Solid-State Circuits*. IEEE, pp. 238–239.
[3] M. Martonosi and D. Clark, "Voltage and Frequency Control With Adaptive Reaction Time in Multiple-Clock-Domain Processors," in *11th International Symposium on High-Performance Computer Architecture*. IEEE, pp. 178–189.
[4] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang, "PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration," *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 445–457, 2014.
[5] M. Powell, S.-H. Y. S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: a circuit technique to reduce leakage in deep-submicron cache memories," *ISLPED00 Proceedings of the 2000 International Symposium on Low Power Electronics and Design Cat No00TH8514*, no. iii, pp. 90–95.
[6] K. Flautner, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *Proceedings 29th Annual International Symposium on Computer Architecture*. IEEE Comput. Soc, pp. 148–157.
[7] A. Basu, D. R. Hower, M. D. Hill, and M. M. Swift, "FreshCache: Statically and dynamically exploiting dataless ways," *2013 IEEE 31st International Conference on Computer Design, ICCD 2013*, pp. 286–293, 2013.
[8] D. H. Albonesi, "Selective cache ways: on-demand cache resource allocation," *MICRO32 Proceedings of the 32nd Annual ACMIEEE International Symposium on Microarchitecture*, pp. 248–259.
[9] K. T. Sundararajan, V. Porpodas, T. M. Jones, N. P. Topham, and B. Franke, "Cooperative partitioning: Energy-efficient cache partitioning for high-performance CMPs," *Proceedings - International Symposium on High-Performance Computer Architecture*, pp. 311–322, 2012.
[10] J. Gaur, M. Chaudhuri, and S. Subramoney, "Bypass and insertion algorithms for exclusive last-level caches," in *Proceedings of the 38th annual international symposium on Computer architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 81–92.
[11] D. Fick, R. G. Dreslinski, B. Giridhar, G. Kim, S. Seo, M. Fojtik, S. Satpathy, Y. Lee, D. Kim, N. Liu, M. Wieckowski, G. Chen, T. Mudge, D. Sylvester, and D. Blaauw, "Centip3De : A 3930DMIPS / W Configurable Near-Threshold 3D Stacked System with," pp. 2011–2013, 2012.
[12] K. Keda, Y. Hagihara, Y. Aimoto, M. Nomura, and Y. Nakazawa, "A Read-Static-Noise-Margin-Free SRAM Cell for," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 113–121, 2006.
[13] M. Qazi, M. E. Sinangil, and A. P. Chandrakasan, "Challenges and Directions for Low-Voltage SRAM," *Design & Test of Computers, IEEE*, vol. 28, no. 1, pp. 32–43, 2011.
[14] Z. Sun, X. Bi, H. H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu, "Multi retention level STT-RAM cache designs with a dynamic refresh scheme," Porto Alegre, Brazil, pp. 329–338, 2011.
[15] M. Sharad, R. Venkatesan, A. Raghunathan, and K. Roy, "Multi-level magnetic RAM using domain wall shift for energy-efficient, high-density caches," *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 64–69, sep 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6629268http://dl.acm.org/citation.cfm?id=2648668.2648685
[16] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE, feb 2011, pp. 50–61. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5749716
[17] C. Zhang, G. Sun, P. Li, and T. Wang, "SBAC: a statistics based cache bypassing method for asymmetric-access caches," *Proceedings of the 2014 . . .*, pp. 345–350.
[18] R. Zhang, K. Wang, B. H. Meyer, M. R. Stan, and K. Skadron, "Architecture implications of pads as a scarce resource," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. IEEE, Jun., pp. 373–384.
[19] Intel, "Intel product information." [Online]. Available: http://ark.intel.com
[20] "International technology roadmap for semiconductors." [Online]. Available: http://www.itrs.net/reports.html
[21] R. Merritt, "Arm cto: power surge could create 'dark silicon'," 2009. [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1172049
[22] B. Bowhill, B. Stackhouse, N. Nassif, Z. Yang, A. Raghavan, C. Morganti, C. Houghton, D. Krueger, O. Franza, J. Desai, J. Crop, D. Bradley, C. Bostak, S. Bhimji, and M. Becker, "4.5 The Xeon processor E5-2600 v3: A 22nm 18-core product family," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*. IEEE, feb, pp. 1–3.
[23] "Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors," 2008.
[24] "Amd turbo core technology." [Online]. Available: http://www.amd.com/en-us/innovations/software-technologies/enhanced-media/turbo-core
[25] D. Skarlatos, R. Thomas, A. Agrawal, S. Qin, R. Pilawa-Podgurski, U. R. Karpuzcu, R. Teodorescu, N. S. Kim, and J. Torrellas, "Snatch: Opportunistically Reassigning Power Allocation between Processor and Memory in 3D Stacks," *International Symposium on Microarchitecture (MICRO)*, 2016.
[26] K. De Vogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "The Energy/Frequency Convexity Rule: Modeling and Experimental Validation on Mobile Devices," in *Parallel Processing and Applied Mathematics (LNCS)*, 2014, pp. 793–803.
[27] H. labs, "Cacti 5.3." [Online]. Available: http://www.hpl.hp.com/research/cacti
[28] S. Bayliss, C. Bouganis, G. Constantinides, and W. Luk, "An fpga implementation of the simplex algorithm," in *Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on*, Dec 2006, pp. 49–56.
[29] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.
[30] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 72–81.
[31] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*. IEEE, 2009, pp. 469–480.
[32] S. L. Xi, H. Jacobson, P. Bose, G.-Y. Wei, and D. Brooks, "Quantifying sources of error in McPAT and potential impacts on architectural studies," 2015, pp. 577–589.
[33] "Intel turbo boost technology 2.0." [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html
[34] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin, "Computational sprinting," *IEEE International Symposium on High-Performance Comp Architecture*, no. Hpca, pp. 1–12, 2012.
[35] M. S. Gupta, J. A. Rivers, P. Bose, G.-Y. Wei, and D. Brooks, "Tribeca:Design for PVT Variations with Local Recovery and Fine-grained Adaptation," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture - Micro-42*. New York, New York, USA: ACM Press, Dec., p. 435.
[36] S. Chen, Y. Hu, Y. Zhang, L. Peng, J. Ardonne, S. Irving, and A. Srivastava, "Increasing off-chip bandwidth in multi-core processors with switchable pins," *ACM SIGARCH Computer Architecture News*, no. 3, pp. 385–396, Oct.